ConEx
Internet-Draft
Intended status: Informational
Expires: August 18, 2014

B. Briscoe
BT

February 14, 2014

Network Performance Isolation using Congestion Policing
draft-briscoe-conex-policing-01

Abstract

This document describes why policing using congestion information can isolate users from network performance degradation due to each other's usage, but without losing the multiplexing benefits of a LAN-style network where anyone can use any amount of any resource. Extensive numerical examples and diagrams are given. The document is agnostic to how the congestion information reaches the policer. The congestion exposure (ConEX) protocol is recommended, but other tunnel feedback mechanisms have been proposed.

Table of Contents

1. Introduction

This document is informative, not normative. It describes why congestion policing isolates the network performance of different parties who are using a network, and why it is more useful than other forms of performance isolation such as peak and committed information rate policing, weighted round-robin or weighted fair-queueing.

The main point is that congestion policing isolates performance even when the load applied by everyone is highly variable, where variation may be over time, or by spreading traffic across different paths (the hose model). Unlike other isolation techniques, congestion policing is an edge mechanism that works over a pool of links across whole network, not just at one link. If the load from everyone happens to be constant, and there is a single bottleneck, congestion policing

gives a nearly identical outcome to weighted round-robin or weighted fair-queuing at that bottleneck. But otherwise, congestion policing is a far more general solution.

The document complements others that describe various network arrangements that could use congestion policing, such as in a broadband access network [I-D.briscoe-conex-initial-deploy], in a mobile access network [I-D.ietf-conex-mobile] or in a data centre network [I-D.briscoe-conex-data-centre]. Nonetheless, all the numerical examples use the data centre scenario.

The key to the solution is the use of congestion-bit-rate rather than bit-rate as the policing metric. _How _this works is very simple and quick to describe Section 2.

However, it is much more difficult to understand _why_ this approach provides performance isolation. In particular, why it provides performance isolation across a network of links, even though there is apparently no isolation mechanism in each link. Section 3 builds up an intuition for why the approach works, and why other approaches fall down in different ways. The bullets below provide a summary of that explanation, which builds from the simple case of long-running flows through a single link up to a full meshed network with on-off flows of different sizes and different behaviours:

- Starting with the simple case of long-running flows focused on a single bottleneck link, tenants get weighted shares of the link, much like weighted round robin, but with no mechanism in any of the links. This is because losses (or ECN marks) are random, so if one tenant sends twice as much bit-rate it will suffer twice as many lost bits (or ECN-marked bits). So, at least for constant long-running flows, regulating congestion-bits gives the same outcome as regulating bits;

- In the more realistic case where flows are not all long-running but a mix of short to very long, it is explained that bit-rate is not a sufficient metric for isolating performance; how _often_ a tenant is sending (or not sending) is the significant factor for performance isolation, not whether bit-rate is shared equally whenever a source happens to be sending;

- Although it might seem that data volume would be a good measure of how often a tenant is sending, we then show why it is not. For instance, a tenant can send a large volume of data but hardly affect the performance of others -- by being more responsive to congestion. Using congestion-volume (congestion-bit-rate over time) in a policer encourages large data senders to be more responsive (to yield), giving other tenants much higher

performance while hardly affecting their own performance. Whereas, using straight volume as an allocation metric provides no distinction between high volume sources that yield and high volume sources that do not yield (the widespread behaviour today);

- We then show that a policer based on the congestion-bit-rate metric works across a network of links treating it as a pool of capacity, whereas other approaches treat each link independently, which is why the proposed approach requires none of the configuration complexity on switches that is involved in other approaches.

- We also show that a congestion policer can be arranged to limit bursts of congestion from sources that focus traffic onto a single link, even where one source may consist of a large aggregate of sources.

- We show that a congestion policer rewards traffic that shifts to less congested paths (e.g. multipath TCP or virtual machine motion).

- We show that congestion policing works on the pool of links, irrespective of whether individual links have significantly different capacities.

- We show that a congestion policer allows a wide variety of responses to congestion (e.g. New Reno TCP [RFC5681], Cubic TCP, Compound TCP, Data Centre TCP [DCTCP] and even unresponsive UDP traffic), while still encouraging and enforcing a sufficient response to congestion from all sources taken together, so that the performance of each application is sufficiently isolated from others.

2. Example Bulk Congestion Policer

In order to explain why a congestion policer works, we first describe a particular design of congestion policer, called a bulk congestion policer. The aim is to give a concrete example to motivate the protocol changes necessary to implement such a policer. This is merely an informative document, so there is no intention to standardise this or any specific congestion policing algorithm. The aim is for different implementers to be free to develop their own improvements to this design.

A number of other documents describe deployment arrangements that include a congestion policer, for instance in a broadband access network [I-D.briscoe-conex-initial-deploy], in a mobile access network [I-D.ietf-conex-mobile] and in a data centre network

[I-D.briscoe-conex-data-centre]. The congestion policer described in the present document could be deployed in any of these arrangements. However, to be concrete, the example link capacities, topology and terminology used in the present document assume a multi-tenant data centre scenario [I-D.briscoe-conex-data-centre]. It is believed that similar examples could be drawn up for the other scenarios, just using different numbers for link capacities, etc and replacing some terminology, e.g. substituting 'end-customer' for 'tenant' throughout.

Figure 1 illustrates a bulk congestion policer. It is very similar to a regular token bucket for policing bit-rate except the tokens do not represent bits, rather they represent 'congestion-bits'. Using Congestion Exposure (ConEx) [I-D.ietf-conex-abstract-mech] is probably the easiest way to understand what congestion-bits are. However, there are other valid ways to signal congestion information to a congestion policer without ConEx (e.g. tunnelled feedback described in [I-D.briscoe-conex-data-centre].

Using ConEx as an example, a ConEx-based congestion token bucket ignores any packets unless they are ConEx-marked. The ConEx protocol includes audit capabilities that force the sender to reveal information it receives about losses anywhere on the downstream path (e.g. at the points marked X in Figure 1). The sender signals this information with a ConEx-marking in the IP header of the packets it sends into the network, using an arrangement called re-inserted feedback or re-feedback. So, every time the sender detects a loss of a 1500B packet, it has to apply a ConEx-marking to a subsequent 1500B packet (or it has to ConEx-mark at least as many bytes made up of smaller packets). In Figure 1 ConEx-marked packets are shown tagged with a '*'.

A bulk congestion policer is deployed at every point where traffic enters an operator's network (similar to the Diffserv architecture). At any one point of entry, the operator assigns a congestion token bucket to each tenant that is sending traffic into the network. The operator assigns a contracted token-fill-rate w1 to tenant 1, w2 to tenant 2, and in general wi to tenant with index i, where i = 1,2,... This contracted token-fill-rate, wi, can be thought of like a weight, where a higher weight allows a tenant to contribute more traffic through a downstream congested link or to contribute traffic into more congested links, as we shall see.

Tenants wanting to send more traffic can be assigned a higher weight (probably paying more for it). Typically, tenants might sign-up to a traffic contract in two parts:

1. a peak rate, which represents the capacity of their access to the network

2. a congestion token-fill rate, which limits how much traffic can be sent that limits what others are sending.

In Figure 1, as each packet arrives, the classifier assigns it to the congestion token bucket of the relevant tenant. The token bucket ignores all non-ConEx-marked packets - they do not drain any tokens from the bucket. Otherwise, if a packet is ConEx-marked and ,say, 1500B in size it will drain 1500B of tokens from the congestion token bucket. If the tenant is contributing to more congestion than its contracted congestion-fill-rate, the bucket level will decrease, and if it persists, the bucket level will become low enough to cause the policer to start discarding a proportion of all that tenant's traffic (whether ConEx-marked or not).

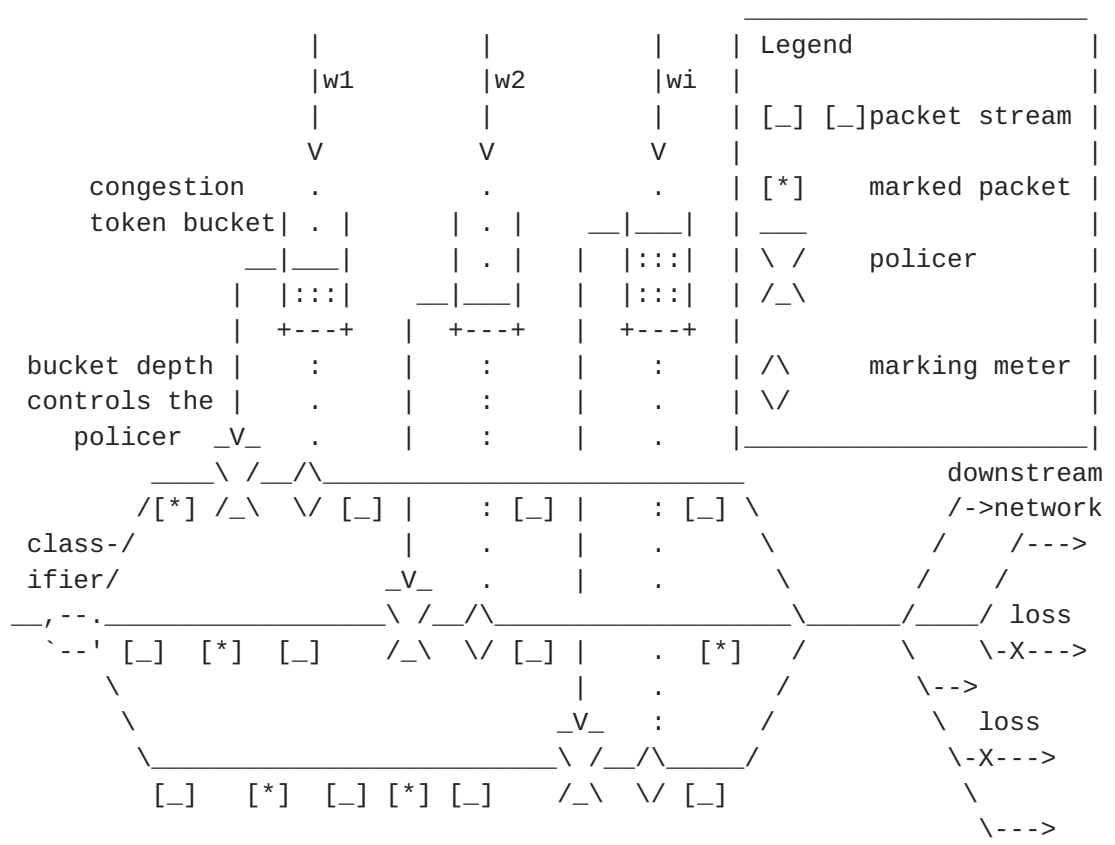

Figure 1: Bulk Congestion Policer Schematic

Note that this is called a bulk congestion policer because it polices all the flows of a tenant even if they spread out in a hose pattern across numerous downstream networks and even if only a few of these flows are primarily to blame for contributing heavily to downstream

congestion, while others find plenty of capacity downstream. This approach is motivated by simplicity rather than precision (if precision were required other policer designs could be used). Simplicity might be more important than precision if capacity were generally well-provisioned and the policer was intended to intervene only rarely to limit unusually extreme behaviour. Also, a network might take such a simple but somewhat heavy-handed approach in order to motivate tenants to implement more complex flow-specific controls themselves [CongPol].

Various more sophisticated congestion policer designs have been evaluated [CPolTrilogyExp]. In these experiments, it was found that it is better if the policer gradually increases discards as the bucket becomes empty. Also isolation between tenants is better if each tenant is policed based on the combination of two buckets, not one (Figure 2):

1. A deep bucket (that would take minutes or even hours to fill at the contracted fill-rate) that constrains the tenant's long-term average rate of congestion (wi)

2. a very shallow bucket (e.g. only two or three MTU) that is filled considerably faster than the deep bucket (c * wi), where c = ~10, which prevents a tenant storing up a large backlog of tokens then causing congestion in one large burst.

In this arrangement each marked packet drains tokens from both buckets, and the probability of policer discard is taken as the worse of the two buckets.

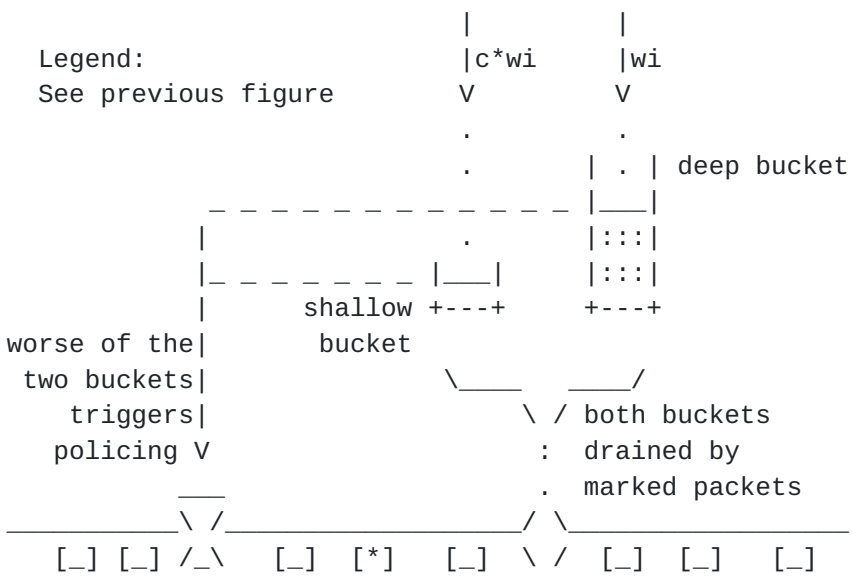

Figure 2: Dual Congestion Token Bucket (in place of each single bucket in the previous figure)

This design of congestion policer will be sufficient to understand the rest of the document. However, it should be remembered that this is only an example, not an ideal design. Also it should be remembered that other mechanisms such as tunnel feedback can be used instead of ConEx.

3. Network Performance Isolation: Intuition

3.1. The Problem

Network performance isolation traditionally meant that each user could be sure of a minimum guaranteed bit-rate. Such assurances are useful if traffic from each tenant follows relatively predictable paths and is fairly constant. If traffic demand is more dynamic and unpredictable (both over time and across paths), minimum bit-rate assurances can still be given, but they have to be very small relative to the available capacity, because a large number of users might all want to simulataneously share any one link, even though they rarely all use it at the same time.

This either means the shared capacity has to be greatly overprovided so that the assured level is large enough, or the assured level has to be small. The former is unnecessarily expensive; the latter doesn't really give a sufficiently useful assurance.

Round robin or fair queuing are other forms of isolation that guarantee that each user will get 1/N of the capacity of each link,

where N is the number of active users at each link. This is fine if the number of active users (N) sharing a link is fairly predictable. However, if large numbers of tenants do not typically share any one link but at any time they all could (as in a data centre), a 1/N assurance is fairly worthless. Again, given N is typically small but could be very large, either the shared capacity has to be expensively overprovided, or the assured bit-rate has to be worthlessly small. The argument is no different for the weighted forms of these algorithms: WRR & WFQ).

Both these traditional forms of isolation try to give one tenant assured instantaneous bit-rate by constraining the instantaneous bit-rate of everyone else. This approach is flawed except in the special case when the load from every tenant on every link is continuous and fairly constant. The reality is usually very different: sources are on-off and the route taken varies, so that on any one link a source is more often off than on.

> In these more realistic (non-constant) scenarios, the capacity available for any one tenant depends much more on _how often_ everyone else uses a link, not just _how much_ bit-rate everyone else would be entitled to if they did use it.

For instance, if 100 tenants are using a 1Gb/s link for 1% of the time, there is a good chance each will get the full 1Gb/s link capacity. But if just six of those tenants suddenly start using the link 50% of the time, whenever the other 94 tenants need the link, they will typically find 3 of these heavier tenants using it already. If a 1/N approach like round-robin were used, then the light tenants would suddenly get 1/4 * 1Gb/s = 250Mb/s on average. Round-robin cannot claim to isolate tenants from each other if they usually get 1Gb/s but sometimes they get 250Mb/s (and only 10Mb/s guaranteed in the worst case when all 100 tenants are active).

In contrast, congestion policing is the key to network performance isolation because it focuses policing only on those tenants that go fast over congested path(s) excessively and persistently over time. This keeps congestion below a design threshold everywhere so that everyone else can go fast. In this way, congestion policing takes account of highly variable loads (varying in time and varying across routes). And, if everyone's load happens to be constant, congestion policing converges on the same outcome as the traditional forms of isolation.

The other flaw in the traditional approaches to isolation, like WRR & WFQ, is that they actually prevent long-running flows from yielding to brief bursts from lighter tenants. A long-running flow can yield to brief flows and still complete nearly as soon as it would have

otherwise (the brief flows complete sooner, freeing up the capacity for the longer flow sooner--see Section 3.7). However, WRR & WFQ prevent flows from even seeing the congestion signals that would allow them to co-ordinate between themselves, because they isolate each tenant completely into separate queues.

In summary, superficially, traditional approaches with separate queues sound good for isolation, but:

1. not when everyone's load is variable, so each tenant has no assurance about how many other queues there will be;

2. and not when each tenant can no longer even see the congestion signals from other tenants, so no-one's control algorithms can determine whether they would benefit most by pushing harder or yielding.

3.2. Approach

It has not been easy to find a way to give the intuition on why congesiton policing isolates performance, particularly across a networks of links not just on a single link. The approach used in this section, is to describe the system as if everyone is using the congestion response they would be forced to use if congestion policing had to intervene. We therefore call this the boundary model of congestion control. It is a very simple congestion response, so it is much easier to understand than if we introduced all the square root terms and other complexity of New Reno TCP's response. And it means we don't have to try to decribe a mix of responses.

The purpose of congestion policing is not to intervene in everyone's rate control all the time. Rather it is to encourage each tenant to _avoid_ being policed -- to keep the aggregate of all their flows' rates within an overall envelope that is sufficiently responsive to congestion. Nonetheless, congestion policing can and will enforce a congestion response if a particular tenant sends traffic that is completely unresponsive to congestion. This upper bound enforced by everyone else's congestion policers is what ensures that each tenant's minimum performance is isolated from the combined effect of everyone else.

We cannot emphasise enough that the intention is not to make individual flows conform to this boundary response to congestion. Indeed the intention is to allow a diverse evolving mix of congestion responses, but constrained in total within a simple overall envelope.

After describing and further justifying using the simple boundary model of congestion control, we start by considering long-running

flows sharing one link. Then we will consider on-off traffic, before widening the scope from one link to a network of links and to links of different sizes. Then we will depart from the initial simplified model of congestion control and consider diverse congestion control algorithms, including completely unresponsive end-systems.

Formal analysis to back-up the intuition provided by this section will be made available in a more extensive companion technical report [conex-dc_tr].

3.3. Terminology

The term congestion probability will be used as a generisation of either loss probability or ECN marking probability.

It is necessary to carefully distinguish:

- congestion bit-rate (or just congestion rate), which is an absolute measure of the rate of congested bits

- congestion probability, which is a relative measure of the proportion of congested bits to all bits

Sometimes people loosely talk of loss rate when they mean loss probability (measured in %). In this document, we will keep to strict terminology, so loss rate would be measured in lost packets per second, or lost bits per second.

3.4. Simple Boundary Model of Congestion Control

The boundary model of congestion control ensures a flow's bit-rate is inversely proportional to the congestion level that it detects. For instance, if congestion probability doubles, the flow's bit-rate halves. This is called a scalable congestion control because it maintains the same rate of congestion signals (marked or dropped packets) no matter how fast it goes. Examples from the research community are Relentless TCP [Relentless] and Scalable TCP [STCP].

In production systems, TCP algorithms based on New Reno [RFC5681] have been widely replaced by alternatives closer to this scalable ideal (e.g. Cubic TCP in Linux and Compound TCP in Windows), because at high rates New Reno generated congestion signals too infrequently to track available capacity fast enough [RFC3649]. More recent TCP updates (e.g. Data Centre TCP [DCTCP] in Windows 8 and available in Linux) are becoming closer still to the scalable ideal.

For instance, consider a scenario where a flow with scalable congestion control is alone in a 1Gb/s link, then another similar

flow from another tenant joins it. Both will push up the congestion probability, which will push down their rates until they together fit into the link. Because the flow's rate has to halve to accomodate the new flow, congestion probability will double (lets say from 0.002% to 0.004%), by our initial assumption of a two similar scalable congestion controls. When it is alone on the link, the congestion-bit-rate of the flow is 20kb/s (=1Gb/s * 0.002%), and when it shares the link it is still 20kb/s (= 500Mb/s * 0.004%).

In summary, a congestion control can be considered scalable if the bit-rate of packets carrying congestion signals (the congestion-bit-rate) always stays the same no matter how much capacity it finds available. This ensures there will always be enough signals in a round trip time to keep the dynamics under control.

Reminder: Making individual flows conform to this boundary (scalable) response to congestion is a non-goal. Although we start this explanation with this specific simple end-system congestion response, this is just to aid intuition.

3.5. Long-Running Flows

Table 1 shows various scenarios where each of five tenants has contracted for 40kb/s of congestion-bit-rate in order to share a 1Gb/s link. In order to help intuition, we start with the (unlikely) scenario where all their flows are long-running. A number of long-running flows will try to use all the link capacity, so for simplicity we take utilisation as a round figure of 100%.

In the case we have just described (scenario A) neither tenant's policer is intervening at all, because both their congestion allowances are 40kb/s and each sends only one flow that contributes 20kb/s of congestion -- half the allowance.

Tenant	contracted congestion-bit-rate kb/s	scenario A # : Mb/s	scenario B # : Mb/s	scenario C # : Mb/s	scenario D # : Mb/s
(a)	40	1 : 500	5 : 250	5 : 200	5 : 250
(b)	40	1 : 500	3 : 250	3 : 200	2 : 250
(c)	40	- : ---	3 : 250	3 : 200	2 : 250
(d)	40	- : ---	2 : 250	2 : 200	1 : 125
(e)	40	- : ---	- : ---	2 : 200	1 : 125
	Congestion probability	0.004%	0.016%	0.02%	0.016%

Table 1: Bit-rates that a congestion policer allocates to five tenants sharing a 1Gb/s link with various numbers (#) of long-running flows all using 'scalable congestion control' of weight 20kb/s

Scenario B shows a case where four of the tenants all send 2 or more long-running flows. Recall, from the assumption of scalable controls, that each flow always contributes 20kb/s no matter how fast it goes. Therefore the policers of tenants (a-c) limit them to two flows-worth of congestion (2 x 20kb/s = 40kb/s). Tenant (d) is only asking for 2 flows, so it gets them without being policed, and all four get the same quarter share of the link.

Scenario C is similar, except the fifth tenant (e) joins in, so they all get equal 1/5 shares of the link.

In Scenario D, only tenant (a) sends more than two flows, so (a)'s policer limits it to two flows-worth of congestion, and everyone else gets the number of flows-worth that they ask for. This means that tenants (d) & (e) get less than everyone else, which is fine because they asked for less than they would have been allowed. (Similarly, in Scenarios A & B, some of the tenants are inactive, so they get zero, which is also less than they could have had if they had wanted.)

With lots of long-running flows, as in scenarios B & C, congestion policing seems to emulate per-tenant round robin scheduling, equalising the bit-rate of each tenant, no matter how many flows they run. By configuring different contracted allowances for each tenant, it can easily be seen that congestion policing could emulate weighted round robin (WRR), with the relative sizes of the allowances acting as the weights.

Scenario D departs from round-robin. This is deliberate, the idea being that tenants are free to take less than their share in the short term, which allows them to take more at other times, as we shall see in Section 3.7. In Scenario D, policing focuses only on the tenant (a) that is continually exceeding its contract. This policer focuses discard solely on tenant a's traffic so that it cannot cause any more congestion at the shared link (shown as 0.016% in the last row).

To summarise so far, ingress congestion policers control congestion-bit-rate in order to indirectly assure a minimum bit-rate per tenant. With lots of long-running flows, the outcome is somewhat similar to WRR, but without the need for any mechanism in each queue.

However, aiming to behave like WRR is only useful when all flows are infinitely long. When load is variable because transfers are of finite size, congestion policing properly isolates one tenant's performance from the behaviour of others, unlike WRR would, as will now be explained.

3.6. On-Off Flows

Figure 3 compares two example scenarios where tenant 'b' regularly sends small files in the top chart and the same size files but more often in the bottom chart (a higher 'on-off ratio'). This is the typical behaviour of a Web server as more clients request more files at peak time. Meanwhile, in this example, tenant c's behaviour doesn't change between the two scenarios -- it sends a couple of large files, each starting at the same time in both cases.

The capacity of the link that 'b' and 'c' share is shown as the full height of the plot. The files sent by 'b' are shown as little rectangles. 'b' can go at the full bit-rate of the link when 'c' is not sending, which is represented by the tall thin rectangles part-way through the trace labelled 'b'. We assume for simplicity that 'b' and 'c' divide up the bit-rate equally. So, when both 'b' and 'c' are sending, the 'b' rectangles are half the height (bit-rate) and twice the width (duration) relative to when 'b' sends alone. The area of a file to be transferred stays the same, whether tall and thin or short and fat, because the area represents the size of the file (bit-rate x duration = file size). The files from 'c' look like inverted castellations, because 'c' uses half the link rate while each file from 'b' completes, then 'c' can fill the link until 'b' starts the next file. The cross-hatched areas represent idle times when no-one is sending.

For this simple scenario we ignore start-up dynamics and just focus on the rate and duration of flows that are long enough to stabilise,

which is why they can be represented as simple rectangles. We will introduce the effect of flow startups later.

In the bottom case, where 'b' sends more often, the gaps between b's transfers are smaller, so 'c' has less opportunity to use the whole line rate. This squeezes out the time it takes for 'c' to complete its file transfers (recall a file will always have the same area which represents its size). Although 'c' finishes later, it still starts the next flow at the same time. In turn, this means 'c' is sending during a greater proprotion of b's transfers, which extends b's average completion time too.

```
  ^ bit-rate
  |
  |---------------------------------------,--.---------,--.----,-------
  |                                       |  |\/\/\/\/\|  |/\/\|
  |                     c                 | b|/\/\/\/\/| b|\/\/|  c
  |------.      ,-----.      ,-----.      |  |\/\/\/\/\|  |/\/\|    ,--
  |  b   |      |  b  |      |  b  |      |  |/\/\/\/\/|  |\/\/|    | b
  |      |      |     |      |     |      |  |\/\/\/\/\|  |/\/\|    |
  +------'------'-----'------'-----'------'--'---------'--'----'----'-->
                                                                    time
  ^ bit-rate
  |
  |---------------------------------------------------.--,--.--,-------
  |                                                   |/\|  |\/|
  |                         c                         |\/| b|/\| c
  |------.  ,-----.  ,-----.  ,-----.  ,-----.  ,-----./\|  |\/|  ,----
  | b    |  | b   |  | b   |  | b   |  | b   |  | b   |\/|  |/\|  | b
  |      |  |     |  |     |  |     |  |     |  |     |/\|  |\/|  |
  +------'--'-----'--'-----'--'-----'--'-----'--'-----'--'--'--'--'---->
                                                                    time
```

Figure 3: In the lower case, the on-off ratio of 'b' has increased, which extends all the completion times of 'c' and 'b'

Round-robin would do little if anything to isolate 'c' from the effect of 'b' sending files more often. Round-robin is designed to force 'b' and 'c' to share the capacity equally when they are both active. But in both scenarios they already share capacity equally when they are both active. The difference is in how often they are active. Round-robin and other traditional fair queuing techniques don't have any memory to sense that 'b' has been active more of the time.

In contrast, a congestion policer can tell when one tenant is sending files more frequently, by measuring the rate at which the tenant is contributing to congestion. Our aim is to show that policers will be

able to isolate performance properly by using the right metric (congestion bit-rate), rather than using the wrong metric (bit-rate), which doesn't sense whether the load over time is large or small.

3.6.1. Numerical Examples Without Policing

The usefulness of the congestion bit-rate metric will now be illustrated with the numerical examples in Table 2. The scenarios illustrate what the congestion bit-rate would be without any policing or scheduling action in the network. Then this metric can be monitored and limited by a policer, to prevent one tenant from harming the performance of others.

The 2nd & 3rd columns (file-size and inter-arrival time) fully represent the behaviour of each tenant in each scenario. All the other columns merely characterise the outcome in various ways. The inter-arrival time (T) is the average time between starting one file and the next. For instance, in scenario E tenant 'b' sends a 16Mb file every 200ms on average. The formula in the heading of some columns shows how the column was derived from other columns.

Scenario E is contrived so that the three tenants all offer the same load to the network, even though they send files of very different size (S). The files sent by tenant 'a' are 100 times smaller than those of tenant 'b', but 'a' sends them 100 times more often. In turn, b's files are 100 times smaller than c's, but 'b' in turn sends them 100 times more often. Graphically, the scenario would look similar to Figure 3, except with three sizes of file, not just two. Scenarios E-G are designed to roughly represent various distributions of file sizes found in data centres, but still to be simple enough to facilitate intuition, even though each tenant would not normally send just one size file.

The average completion time (t) and the maximum were calculated from a fairly simple analytical model (documented in a campanion technical report [conex-dc_tr]). Using one data point as an example, it can be seen that a 1600Mb (200MB) file from tenant 'c' completes in 1905ms (about 1.9s). The files that are 100 times smaller complete 100 times more quickly on average. In fact, in this scenario with equal loads, each tenant perceives that their files are being transferred at the same rate of 840Mb/s on average (file-size divided by completion time, as shown in the 'apparent bit-rate' column). Thus on average all three tenants perceive they are getting 84% of the 1Gb /s link on average (due to the benefit of multiplexing and utilisation being low at 240Mb/s / 1Gb/s = 24% in this case).

The completion times of the smaller files vary significantly, depending on whether a larger file transfer is proceeding at the same

time. We have already seen this effect in Figure 3, where, when tenant b's files share with 'c', they take twice as long to complete as when they don't. This is why the maximum completion time is greater than the average for the small files, whereas there is imperceptible variance for the largest files.

The final column shows how congestion bit-rate will be a useful metric to enforce performance isolation (as we said, the table illustrates the situation before any enforcement mechanism is added). In the case of equal loads (scenario E), average congestion bit-rates are all equal. In scenarios F and G average congestion bit-rates are higher, because all tenants are placing much more load on the network over time, even though each still sends at equal rate to others when they are active together. Figure 3 illustrated a similar effect in the difference between the top and bottom scenarios.

The maximum instantaneous congestion bit-rate is nearly always 20kb/s. That is because, by definition, all the tenants are using scalable congestion controls with a constant congestion rate of 20kb/s, and each tenant's flows rarely overlap because the per-tenant load in these scenarios is fairly low. As we saw in Section 3.4, the congestion rate of a particular scalable congestion control is always the same, no matter how many other flows it competes with.

Once it is understood that the congestion bit-rate of one scalable flow is always 'w' and doesn't change whenever a flow is active, it becomes clear what the congestion bit-rate will be when averaged over time; it will simply be 'w' multiplied by the proportion of time that the tenant's file transfers are active. That is, w*t/T. For instance, in scenario E, on average tenant b's flows start 200ms apart, but they complete in 19ms. So they are active for 19/200 = 10% of the time (rounded). A tenant that causes a congestion bit-rate of 20kb/s for 10% of the time will have an average congestion-bit-rate of 2kb/s, as shown.

To summarise so far, no matter how many more files other tenants transfer at the same time, each scalable flow still contributes to congestion at the same rate, but it contributes for more of the time, because it squeezes out into the gap before the next flow from that tenant starts.

Tenant	File size S Mb	Ave. inter-arr-ival T ms	Ave. load S/T Mb/s	Completion time ave : max ms	Apparent bit-rate ave : min S/t Mb/s	Congestion bit-rate ave : max w*t/T kb/s
				Scenario E		
a	0.16	2	80	0.19 : 0.48	840 : 333	2 : 20
b	16	200	80	19 : 35	840 : 460	2 : 20
c	1600	20000	80	1905 : 1905	840 : 840	2 : 20
			240			
				Scenario F		
a	0.16	0.67	240	0.31 : 0.48	516 : 333	9 : 20
b	16	50	320	29 : 42	557 : 380	11 : 20
c	1600	10000	160	3636 : 3636	440 : 440	7 : 20
			720			
				Scenario G		
a	0.16	0.67	240	0.33 : 0.64	481 : 250	10 : 20
b	16	40	400	32 : 46	505 : 345	16 : 40
c	1600	10000	160	4543 : 4543	352 : 352	9 : 20
			800			

Single link of capacity 1Gb/s. Each tenant uses a scalable congestion control which contributes a congestion-bit-rate for each flow of w = 20kb/s.

Table 2: How the effect on others of various file-transfer behaviours can be measured by the resulting congestion-bit-rate

In scenario F, clients have increased the rate they request files from tenants a, b and c respectively by 3x, 4x and 2x relative to scenario E. The tenants send the same size files but 3x, 4x and 2x more often. For instance tenant 'b' is sending 16Mb files four times as often as before, and they now take longer as well -- nearly 29ms rather than 19ms -- because the other tenants are active more often too, so completion gets squeezed to later. Consequently, tenant 'b' is now sending 57% of the time, so its congestion-bit-rate is 20kb/s * 57% = 11kb/s. This is nearly 6x higher than in scenario E, reflecting both b's own increase by 4x and that this increase coincides with everyone else increasing their load.

In scenario G, tenant 'b' increases even more, to 5x the load it offered in scenario E. This results in average utilisation of 800Mb/s / 1Gb/s = 80%, compared to 72% in scenario F and only 24% in scenario E. 'b' sends the same files but 5x more often, so its load rises 5x.

Completion times rise for everyone due to the overall rise in load, but the congestion rates of 'a' and 'c' don't rise anything like as much as that of 'b', because they still leave large gaps between files. For instance, tenant 'c' completes each large file transfer in 4.5s (compared to 1.9s in scenario E), but it still only sends files every 10s. So 'c' only sends 45% of the time, which is reflected in its congestion bit-rate of 20kb/s * 45% = 9kb/s.

In contrast, on average tenant 'b' can only complete each medium-sized file transfer in 32ms (compared to 19ms in scenario E), but on average it starts sending another file after 40ms. So 'b' sends 79% of the time, which is reflected in its congestion bit-rate of 20kb/s * 79% = 16kb/s (rounded).

However, during the 45% of the time that 'c' sends a large file, b's completion time is higher than average (as shown in Figure 3). In fact, as shown in the maximum completion time column, 'b' completes in 46ms, but it starts sending a new file after 40ms, which is before the previous one has completed. Therefore, during each of c's large files, 'b' sends 46/40 = 116% of the time on average.

This actually means 'b' is overlapping two files for 16% of the time on average and sending one file for the remaining 84%. Whenever two file transfers overlap, 'b' will be causing 2 x 20kb/s = 40kb/s of congestion, which explains why tenant b in scenario G is the only case with a maximum congestion rate of 40kb/s rather than 20kb/s as in every other case. Over the duration of c's large files, 'b' would therefore cause congestion at an average rate of 20kb/s * 84% + 40kb/s * 16% = 23kb/s (or more simply 20kb/s * 116% = 23kb/s). Of course, when 'c' is not sending a large file, 'b' will contribute less to congestion, which is why its average congestion rate is 16kb/s overall, as discussed earlier.

3.6.2. Congestion Policing of On-Off Flows

Still referring to the numerical examples in Table 2, we will now discuss the effect of limiting each tenant with a congestion policer.

The network operator might have deployed congestion policers to cap each tenant's average congestion rate to 16kb/s. None of the tenants are exceeding this limit in any of the scenarios, but tenant 'b' is just shy of it in scenario G. Therefore all the tenants would be free to behave in all sorts of ways, such as those of scenarios E-G.

However, they would be prevented from degrading the performance of the other tenants beyond the point reached by tenant 'b' in scenario G. If tenant 'b' added more load, the policer would prevent the extra load entering the network by focusing drop solely on tenant 'b', preventing the other tenants from experiencing any more congestion due to tenant 'b'. Then tenants 'a' and 'c' would be assured the (average) apparent bit-rates shown, whatever the behaviour of 'b'.

If 'a' added more load, 'c' would not suffer. Instead 'b' would go over limit and its rate would be trimmed during congestion peaks, sacrificing some of its lead to 'a'. Similarly, if 'c' added more load, 'b' would be made to sacrifice some of its performance, so that 'a' would not suffer. Further, if more tenants arrived to share the same link, the policer would force 'b' to sacrifice performance in favour of the additional tenants.

There is nothing special about a policer limit of 16kb/s. The example when discussing infinite flows used a limit of 40kb/s per tenant. And some tenants can be given higher limits than others (e.g. at an additional charge). If the operator gives out congestion limits that together add up to a higher amount but it doesn't increase the link capacity, it merely allows the tenants to apply more load (e.g. more files of the same size in the same time), but each with lower bit-rate.

3.7. Weighted Congestion Controls

At high speed, congestion controls such as Cubic TCP, Data Centre TCP, Compound TCP etc all contribute to congestion at widely differing rates, which is called their 'aggressiveness' or 'weight'. So far, we have made the simplifying assumption of a scalable congestion control algorithm that contributes to congestion at a constant rate of w = 20kb/s. We now assume tenant 'c' uses a similar congestion control to before, but with different parameters in the algorithm so that its weight is still constant, but much lower, at w = 2.2kb/s.

Tenant 'b' is sending smaler files than 'c', so it still uses w = 20kb/s. Then, when the two compete for the 1Gb/s link, they will share it in proportion to their weights, 20:2.2 (or 90%:10%). That is, when competing, 'b' and 'c' will respectively get (20/22.2)*1Gb/s = 900Mb/s and (2.2/22.2)*1Gb/s = 100Mb/s of the 1Gb/s link. Figure 4 shows the situation before (upper) and after (lower) this change.

When the two compete, 'b' transfers each file 9/5 faster than before (900Mb/s rather than 500Mb/s), so it completes them in 5/9 of the time. 'b' still contributes congestion at the same rate of 20kb/s, but for 5/9 less time than before. Therefore, relative to before,

'b' uses up its allowance only just over half as quickly, and it completes each transfer nearly twice as fast.

Even though 'b' completes each small file much faster, 'c' should complete its file transfer hardly any later than before. Although tenant 'b' goes faster, as each 'b' file finishes, it gets out of the way sooner. Irrespective of its lower weight, 'c' can still use the full link when 'b' is not present, because weights only determine shares between active flows. Because 'c' has the whole link sooner it can catch up to the same point it would have reached in its download if 'b' had been slower but longer. Tenant 'c' will probably lose some completion time because it has to accelerate and decelerate more. But, whenever it is sending a file, 'c' gains (20kb/s -2kb/s) = 18kb of allowance every second, which it can use for other transfers.

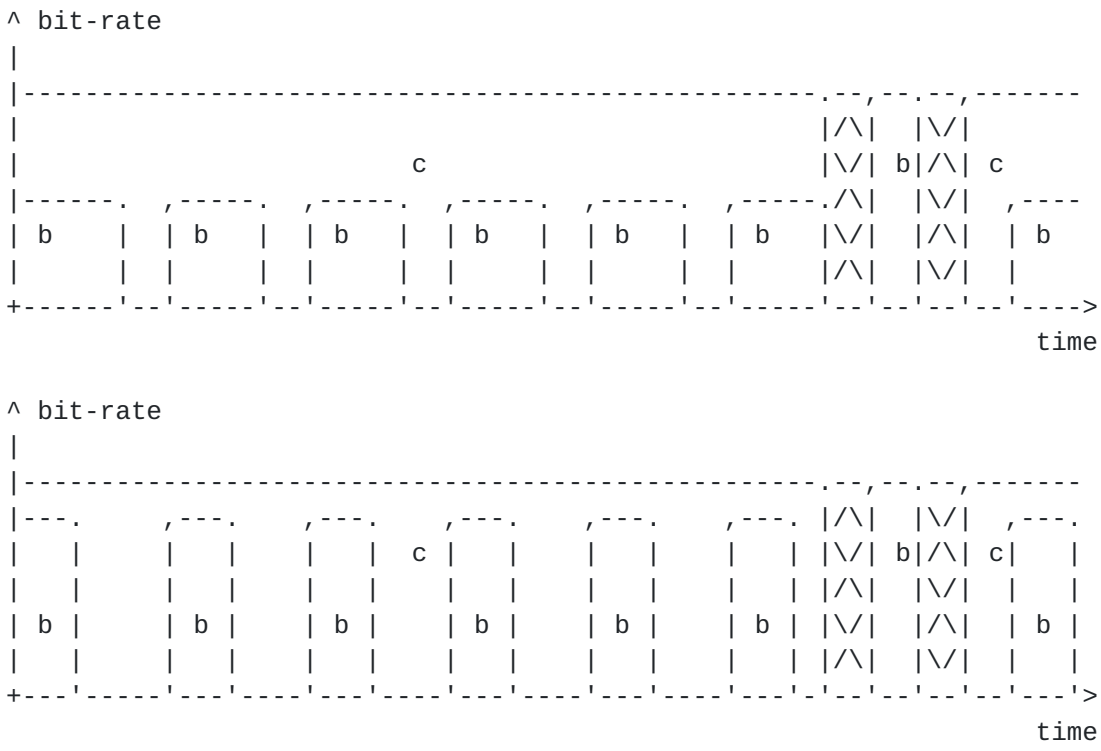

Figure 4: Weighted congestion controls with equal weights (upper) and unequal (lower)

It seems too good to be true that both tenants gain so much and lose so little by 'c' reducing its aggressiveness. Effectively 'c' allows everyone else's shorter flows to proceed as if the long flows were hardly there, and 'c' hardly loses out at all itself. The gains are unlikely to be as perfect as this simple model predicts, but we believe they will be nearly as substantial.

It might seem that everyone can keep gaining by everyone agreeing to reduce their weights, ad infinitum. However, the lower the weight, the less signals the congestion control gets, so it starts to lose its control during dynamics. Nonetheless, congestion policing should encourage congestion control designs to keep reducing their weights, but they will have to stop when they reach the minimum necessary congestion in order to maintain sufficient control signals.

3.8. A Network of Links

So far we have only considered a single link. Congestion policing at the network edge is designed to work across a network of links, treating them all as a pool of resources, as we shall now explain. We will use the dual-homed data centre topology shown in Figure 5 (stretching the bounds of ASCII art) as a simple example of a pool of resources.

In this case where there are 48 servers (H1, H2, ... Hn where n=48) on the left, with on average 8 virtual machines (VMs) running on each (e.g. server n is running Vn1, Vn2, ... to Vnm where m = 8). Each server is connected by two 1Gb/s links, one to each top-of-rack switch S1 & S2. To the right of the switches, there are 6 links of 10Gb/s each, connecting onwards to customer networks or to the rest of the data centre. There is a total of 48 *2 *1Gb/s = 96Gb/s capacity between the 48 servers and the 2 switches, but there is only 6 *10Gb/s = 60Gb/s to the right of the switches. Nonetheless, data centres are often designed with some level of contention like this, because at the ToR switches a proportion of the traffic from certain hosts turns round locally towards other hosts in the same rack.

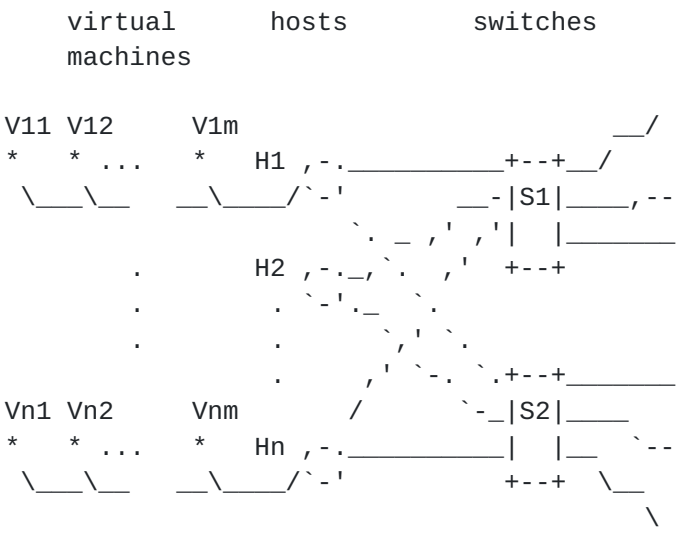

Figure 5: Dual-Homed Topology -- a Simple Resource Pool

The congestion policer proposed in this document is based on the 'hose' model, where a tenant's congestion allowance can be used for sending data over any path, including many paths at once. Therefore, any one of the virtual machines on the left can use its allowance to contribute to congestion on any or all of the 6 links on the right (or any other link in the diagram actually, including those from the server to the switches and those turning back to other hosts).

Nonetheless, if congestion policers are to enforce performance isolation, they should stop one tenant squeezing the capacity available to another tenant who needs to use a particular bottleneck link or links (e.g. to reach a particular receiver). Policing should work whether the offending tenant is acting deliberately or merely carelessly.

3.8.1. Numerical Example: Isolation from Focused Load

Consider the following scenario: two tenants have equal congestion allowances of 40kb/s, and let's imagine they each run servers that send about 1 flow per second. All their flows use the same scalable congestion control and all use the same aggressiveness or weight of 20kb/s.

Tenant A has clients all over the network, so it spreads its traffic over numerous links, and its flows usually complete within 1 second. Therefore, on average, A has one flow running at any one instant, so it consumes 20kb/s from its 40kb/s congestion allowance.

Then tenant B decides (carelessly or deliberately) to concentrate all its flows into one of the links that A sometimes uses for one of its flows. All the flows through this new bottleneck (A's, B's and those from other tenants) will still each consume 20kb/s of allowance while they are active (Section 3.6), but they will all take longer to complete. Let's say they take 15 times longer to complete on average (i.e. 15s not 1s). And let's assume 5% of tenant A's flows pass through this link. Then each tenant will consume congestion allowance at:

- Tenant A: (95% * 1 * 20kb/s) + (5% * 15 * 20kb/s) = 34kb/s

- Tenant B: (100% * 15 * 20kb/s) = 300kb/s

Tenant B's congestion token bucket will therefore drain 7.5 times faster than it is filling (300 / 40 = 7.5) so it will rapidly empty and start dropping some of tenant B's traffic before it enters the network. In fact, once tenant B's bucket is empty, it will be limited to no more than 40kb/s of congestion at the bottleneck link, rather than 300kb/s otherwise. In effect, the policer shifts packet

drops in excess of the 40kb/s allowance out of the shared link an into itself, to focus excess discards only on tenant B.

Once tenant B's bucket empties, tenant A's completion time for the flows through this bottleneck would not return to 1s, but it would be limited to about 2s (not 15s), and it would consume its congestion allowance at about 21kb/s, well within its limt of 40kb/s.

3.8.2. Isolation in the Short-Term

It seems problematic that tenant B's congestion policer takes time to empty before it kicks in, during which time other tenants suffer reduced performance. This is deliberate--to allow some give and take over time. Nonetheless, it is possible to limit this problem as well, using the dual token bucket already described in Figure 5. The second shallow bucket sets an immediate limit on how much tenant B can harm the performance of others. The fill-rate of the shallow bucket is c times that of the deeper bucket (let's say c = 8), so tenant B is immediately limited to 8 * 40kb/s = 320kb/s of congestion. In the scenario described above, tenant B only reaches 300kb/s but at least this backstop prevents congestion from tenant B ever exceeding 320kb/s.

3.8.3. Encouraging Load Balancing

A policer may not even have to directly intervene for tenants to be protected; it might encourage load balancing to remove the problem first. Load balancing might either be provided by the network (usually just random), or some of the tenants might themselves actively shift traffic off the increasingly congested bottleneck and onto other paths. Some of them might be using the multipath TCP protocol (MPTCP -- see experimental [RFC6356]) that would achieve this automatically, or ultimately if the congestion signals persisted, automatic VM placement algorithms might shift their virtual machine to a different endpoint to circumvent the congestion hot-spot completely. Even if some tenants were not using MPTCP or could not shift easily, others shifting away would achieve the same outcome. Essentially, the deterrent effect of congestion policers encourages everyone to even out congestion across the network, shifting load away from hot spots. Then performance isolation becomes an emergent property of everyone's behaviour, due to the deterrent effect of policers, rather than always through explicit policer intervention.

In such an environment, a policer is needed that shares out the whole pool of network resources, not one that just controls shares of each link individually.

In contrast, enforcement mechanisms based on scheduling algorithms like WRR or WFQ have to be deployed at each link, and each one works in ignorance of how much of other links the tenant is using. These algorithms were designed for networks with a single known bottleneck per customer (e.g. many access networks). However, in data centres there are many potential bottlenecks and each tenant generally only uses a share of a small number of them. A mechanism like WRR would not isolate anyone's performance if it gave every tenant the right to use the same share of all the links in the network, without regard to how many they were using.

3.9. Tenants of Different Sizes

The scenario in Section 3.8 assumed tenants A & B had equal congestion allowances. If one tenant bought a huge allowance (e.g. 500kb/s) while another tenant bought a small allowance (e.g. 10kb/s), it would seem that the former could focus all its traffic on one link to swamp the latter.

At this point, it needs to be pointed out that congestion allowances do not preclude the need for decent capacity planning. If we further assume that tenant A has bought the same access capacity as tenant B, 50 times more congestion allowance implies tenant A expects to be provided with 50 times the contention ratio of tenant B, i.e. any single links they share will be big enough for B to still have enough if it gets 1 share for every 50 used by A.

However, there is still a problem in that A can store up congestion allowance and cause much more than 500kb/s of congestion in one burst. The dual token bucket arrangement (Figure 5) limits tenant A to c * 500kb/s of congestion, but if c=8 as before, 8 * 500kb/s = 4Mb /s of congestion in a burst would seriously swamp tenant B if all focused on one link where tenant B was quietly consuming its 10kb/s allowance.

The answer to this problem is that the burst limit c can be smaller for larger tenants, tending to 1 for the largest tenants (i.e. a single shallow bucket would suffice for huge tenants). The reasoning is that the c factor allows a tenant some give-and-take over time, but the aggregate of traffic from a larger tenant consists of enough flows that it has sufficient give-and-take within its own aggregate without needing give-and-take from others.

3.10. Links of Different Sizes

Congestion policing treats a Mb/s of capacity in one link as identical to a Mb/s of capacity in another link, even if the size of each link is different. For instance, consider the case where one of

the three links to the right of each switch in Figure 5 were upgraded to 40Gb/s while the other two remained at 10Gb/s (perhaps to accommodate the extra traffic from a couple of the dual homed 1Gb/s servers being upgraded to dual-homed 10Gb/s).

A congestion control algorithm running at the same rate will cause the same level of congestion probability, whatever size link it is sharing. For example:

- If 50 equal flows share a 10Gb/s link (10Gb/s / 50 = 200Mb/s each) they will cause 0.01% congestion probability;

- If 200 of the same flows share a 40Gb/s link (40Gb/s / 200 = 200Mb/s each) they will still cause 0.01% congestion probability;

This is because the congestion probability is determined by the congestion control algorithms, not by the link.

Therefore, if an average of 300 flows were spread across the above links (1x 40Gb/s and 2 x 10Gb/s), the numbers on each link would tend towards respectively 200:50:50, so that each flow would get 200Mb/s and each link would have 0.01% congestion on it.

Sometimes, there might be more flows on one link, resulting in less than 200Mb/s per flow and congestion higher than 0.01%. However, whenever the congestion level was greater on one link than another, congestion policing would encourage flows to balance out the congestion level across the links (as long as some flows could use congestion balancing mechanisms like MPTCP).

In summary, all the outcomes of congestion policing described so far (emulating WRR, etc) apply across a pool of diverse link sizes just as much as they apply to single links, without any need for the links to signal to each other nor to a central controller.

3.11. Diverse Congestion Control Algorithms

Throughout this explanation we have assumed a scalable congestion control algorithm, which we justified (Section 3.4) as the 'boundary' case if congestion policing were to intervene, which is all that is relevant when considering whether the policer can enforce performance isolation.

The defining difference between the scalable congestion we have assumed and the congestion controls in widespread production operating systems (New Reno, Compound, Cubic, Data Centre TCP etc) is the way congestion probability decreases as flow-rate increases (for a long-running flow). With a scalable congestion control, if flow-

rate doubles, congestion probability halves. Whereas, with most production congestion controls, if flow-rate doubles, congestion probability reduces to less than half. For instance, New Reno TCP reduces congestion to a quarter. The responses of Cubic and Compound are closer to the ideal scalable control than to New Reno, but they do not depart too far from New Reno to ensure they can co-exist happily with it.

Congestion policing still works, whether or not the congestion controls in daily use by tenants fit the scalable model. A bulk congestion policer constrains the sum of all the congestion controls being used by a tenant so that they collectively remain below an aggregate envelope that is itself shaped like the sum of many scalable algorithms. Bulk congestion policers will constrain the overall congestion effect (the sum) of any mix of algorithms within it, including flows that are completely unresponsive to congestion.

This will now be explained using Figure 6 (if the ASCII art is incomprehensible, see a similar plot at Figure 3 of [CongPol]).

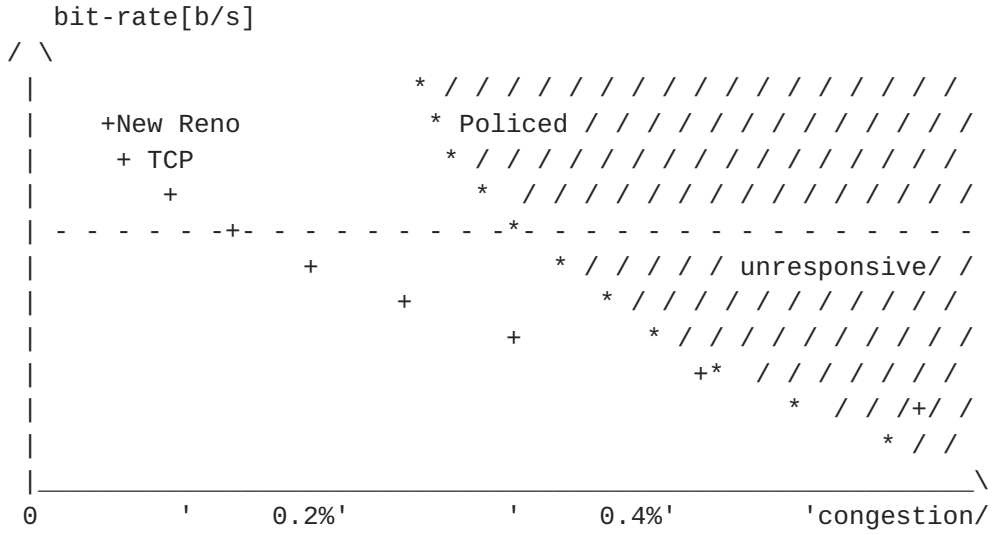

Figure 6: Schematic Plot of Bulk Policing of Traffic with Different Congestion Responses

The congestion policer prevents the aggregate of a tenant's traffic from operating in the hatched region shown in Figure 6 (there is deliberately no vertical scale, because the plot is schematic only). The tenant can have either high congestion or high rate, but not both at the same time. It can be seen that the single New Reno TCP flow responds to congestion in a similar way, but with a flatter slope. It can also be seen that, at 0.25% prevailing congestion, this policer would allow roughly two simultaneous New Reno TCP flows without intervening. At about 0.45% congestion it would allow only

one, and above 0.45% it would start to police even one flow. But at much lower congestion levels typical in a high-capacity data centre (e.g. 0.01%) the policer would allow numerous New Reno TCP flows (extrapolating the curve upwards).

The horizontal plot represents an unresponsive (UDP) flow. The congestion policer would allow one of these flows to run unimpeded up to roughly 0.3% congestion, when the policer would effectively take control and impose its own congestion response on this flow, dropping more UDP packets for higher levels of prevailing congestion. If a tenant ran two of these UDP flows (or one at twice the rate), the policer would leave them alone unless prevailing congestion was above about 0.22% when it would intervene (by extrapolation of the visible plots).

The New Reno TCP curve follows the characteristic k/sqrt(p) rule, while the 'Policed' curve follows a simple w/p rule, where k is the New Reno constant and w is the constant contracted fill-rate of a particular policer. A scalable TCP (not shown) would follow a v/p rule where v is the weight of the individual flow. If we imagine a scenario where w = 8v, the policer would allow 8 of these scalable TCP flows (averaged over time), and this would be true at any congestion level, because the curvature of 8 scalable TCP's would exacatly match the 'Policed' curve.

So far, the discussion has focused on the congestion avoidance phase of each congestion response. As each TCP flow starts, it typically increases exponentially (TCP slow-start) until it overshoots available capacity causing a spike of loss due to the feedback delay over the following round-trip. The ConEx audit mechanism requires the source to send credit markings in advance to cover the risk of these spikes of loss. Credit marked ConEx packets drain tokens from the congestion policer similarly to regular ConEx marked packets, which reduces the available congestion allowance for other flows. If the tenant starts new flows fast enough, these credit markings alone will empty the token bucket so that the policer starts to discard some packets from the start-up phase before they can enter the network. (The tunnel feedback method for signalling congestion back to the policer lacks such a credit mechanism, which is an important advantage of the ConEx protocol, given Internet traffic generally consists of a high proportion of short flows.)

A tenant could run a mix of multiple different types of TCP and UDP flows, as long as, when all stacked up, the sum of them all still fitted under the 'Policed' curve at the prevailing level of congestion.

4. Parameter Setting

The primary parameter to set for each tenant is the contracted fill-rate of their congestion token bucket. For a server sending predominantly long-running flows (e.g.a video server or an indexing robot filling search databases), this fill-rate can be determined from the required number of simultaneous TCP flows, and the typical congestion-bit-rate of one TCP flow.

For a scalable TCP flow, the congestion bit-rate is constant. But currently (2013) TCP Cubic is predominant, and the congestion bit-rate of one Cubic flow is not constant, but depends somewhat on bit-rate and RTT. For example, a Cubic flow with 100Mb/s throughput and 50ms RTT, has a congestion-bit-rate of ~2kbs, whereas at 200Mb/s and 5ms RTT, its congestion-bit-rate is ~6kb/s. Therefore, a reasonable rule of thumb for Cubic's congestion-bit-rate at rates and RTTs of about this order is 5kb/s.

Therefore, if a tenant wanted to serve no more on average than 12 simultaneous TCP Cubic flows, their contracted congestion-rate should be roughly 12 * 5kb/s = 60kb/s.

The contracted fill rate of a congestion policer should not need to change as flow-rates and link capacities scale, assuming the tenant still utilises the higher capacity to the same extent.

A congestion policer based on the dual token bucket in Figure 2 also needs to specify the c factor that determined the maximum congestion-rate allowed, rather than the average. A theoretical approach to setting this factor is being worked on but, in the mean time, a figure of about 10 seems reasonable for a tenant sending only a few simultaneous flows (see Section 3.8.2), while Section 3.9 explains that the c factor should tend to reduce down to 1 for very large tenants.

The dual token bucket makes isolation fairly insensitive to the depth of the deeper bucket, because the shallower bucket constrains the congestion burst rate, which makes the congestion burst size less important. It is believed that the deeper bucket depth could be minutes or even hours of token fill. The depth of the shallower bucket should be just a few MTU--more experimentation is needed to determine how few.

Indeed, all the above recommendations on parameter setting are best considered as initial values to be used in experiments to determine whether other values would be better.

No recommendations can yet be given for parameter settings for a tenant running a large proportion of short flows. This requires further experimentation.

5. Security Considerations

This document is about performance isolation, therefore the whole document concerns security. Where ConEx is used, the specification of the ConEx Abstract Mechanism [I-D.ietf-conex-abstract-mech] should be consulted for security matters, particularly the design of audit to assure the integrity of ConEx signals.

6. IANA Considerations

{Section to be removed by the RFC Editor}. This document does not require actions by IANA.

7. Conclusions

{ToDo}

8. Acknowledgments

Bob Briscoe is part-funded by the European Community under its Seventh Framework Programme through the Trilogy 2 project (ICT-317756). The views expressed here are solely those of the author.

9. Informative References

[CPolTrilogyExp] Raiciu, C., Ed., "Progress on resource control", Trilogy EU 7th Framework Project ICT-216372 Deliverable 9, December 2009, <http://trilogy-project.org/publications/deliverables.html>.

[CongPol] Jacquet, A., Briscoe, B., and T. Moncaster, "Policing Freedom to Use the Internet Resource Pool", Proc ACM Workshop on Re-Architecting the Internet (ReArch'08) , December 2008, <http://bobbriscoe.net/projects/refb/#polfree>.

[DCTCP] Alizadeh, M., Greenberg, A., Maltz, D., Padhye, J., Patel, P., Prabhakar, B., Sengupta, S., and M. Sridharan, "Data Center TCP (DCTCP)", ACM SIGCOMM CCR 40(4)63--74, October 2010, <http://portal.acm.org/citation.cfm?id=1851192>.

[I-D.briscoe-conex-data-centre]
Briscoe, B. and M. Sridharan, "Network Performance Isolation in Data Centres using Congestion Policing", draft-briscoe-conex-data-centre-01 (work in progress), February 2013.

[I-D.briscoe-conex-initial-deploy]
Briscoe, B., "Initial Congestion Exposure (ConEx) Deployment Examples", draft-briscoe-conex-initial-deploy-03 (work in progress), July 2012.

[I-D.ietf-conex-abstract-mech]
Mathis, M. and B. Briscoe, "Congestion Exposure (ConEx) Concepts and Abstract Mechanism", draft-ietf-conex-abstract-mech-08 (work in progress), October 2013.

[I-D.ietf-conex-mobile]
Kutscher, D., Mir, F., Winter, R., Krishnan, S., Zhang, Y., and C. Bernardos, "Mobile Communication Congestion Exposure Scenario", draft-ietf-conex-mobile-03 (work in progress), February 2014.

[RFC3649] Floyd, S., "HighSpeed TCP for Large Congestion Windows", RFC 3649, December 2003.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, September 2009.

[RFC6356] Raiciu, C., Handley, M., and D. Wischik, "Coupled Congestion Control for Multipath Transport Protocols", RFC 6356, October 2011.

[Relentless]
Mathis, M., "Relentless Congestion Control", Proc. Int'l Wkshp on Protocols for Future, Large-scale & Diverse Network Transports (PFLDNeT'09) , May 2009, <http://www.hpcc.jp/pfldnet2009/Program.html>.

[STCP] Kelly, T., "Scalable TCP: Improving Performance in Highspeed Wide Area Networks", ACM SIGCOMM Computer Communication Review 32(2), April 2003, <http://www-lce.eng.cam.ac.uk/~ctk21/scalable/>.

[conex-dc_tr]
Briscoe, , "Network Performance Isolation in Data Centres by Congestion Exposure to Edge Policers", BT Technical Report TR-DES8-2011-004, November 2011.

To appear

Appendix A. Summary of Changes between Drafts

From briscoe-00 to briscoe-01: No technical differences; merely clarified throughout & updated references.

From briscoe-conex-data-centre-00 to briscoe-conex-policing-00: This draft was separated out from the -00 version of [I-D.briscoe-conex-data-centre], taken mostly from what was section 4. Changes from that draft are listed below:

From section 4 of draft-briscoe-conex-data-centre-00:

+ New Introduction

+ Added section 2. "Example Bulk Congestion Policer"

+ Rearranged and clarified the previous introductory text that preceded what is now Section 3.4 into three subsections:

  3.1 "The Problem"

  3.2 "Approach"

  3.3 "Terminology"

+ Corrected numerical examples:

  - Section 3.4 "Long-RunningFlows": 0.04% to 0.004% and 400kb/s to 40kb/s

  - Section 3.6.1 "Numerical Examples Without Policing": 10kb/s to 20kb/s

  - section 3.7 "Weighted Congestion Controls": 22.2 to 20

+ Clarified where necessary, especially the descriptions of the scenarios in 3.7 "Weighted Congestion Controls" and 3.8 "A Network of Links".

+ Section 3.8 "A Network of Links": added a numerical example including congestion burst limiting that was not covered at all previously

+ Added Section 3.9 "Tenants of Different Sizes"

+ Filled in absent text in:

  - Section 3.11 "Diverse Congestion Controls"

- Section 4. "Parameters Settings"

- Section 5. "Security Considerations"

+ Minor corrections throughout and updated references.

Author's Address

Bob Briscoe
BT
B54/77, Adastral Park
Martlesham Heath
Ipswich IP5 3RE
UK

Phone: +44 1473 645196
EMail: bob.briscoe@bt.com
URI: http://bobbriscoe.net/